# Modeling Machine Learning: A Cognitive Economic Approach*

Andrew Caplin†, Daniel Martin‡, and Philip Marx§

January 25, 2024

**Abstract**

We apply methodological innovations from cognitive economics that were designed to study human cognition to instead better understand machine learning. We first show that the folk theory of machine learning – that an algorithms learns optimally to minimize the loss function used in training – rests on a shaky foundation. We then identify a path forward by translating ideas from the costly learning branch of cognitive economics. We find that changes in the loss function impact learning just as they might if the algorithm was a rational human being who found learning costly according to a revealed *pseudo-cost* function that may or may not correspond to actual resource costs. Our approach can be leveraged to determine more effective loss functions given a third party's objective, be it a firm or a policy maker.

*We thank Tai-Wei Hu, Emir Kamenica, Stephen Morris, Nick Netzer, Marek Pycia, Doron Ravid, Jakub Steiner, Colin Stewart, and audiences at Zurich, ESSET, the Sloan-NOMIS Summer School on Cognitive Foundation of Economic Behavior, D-TEA, and SAET. We thank the Alfred P. Sloan Foundation for Grant G-2023-19660 supporting our research on Cognitive Economics at Work. We are also grateful to the LSU High Performance Computing and Northwestern Research Computing Services for providing computational resources.

†Department of Economics, New York University.

‡Department of Economics, University of California, Santa Barbara and Kellogg School of Management, Northwestern University.

§Department of Economics, Louisiana State University.

# 1 Introduction

Artificial intelligence (AI), particularly machine learning, has become increasingly central to many of the most important decisions we make.[1] In this paper we focus on a key link in the human-AI decision-making pipeline, that of model training. This is the stage at which data scientists generalize from ground truth instances to produce a trained model that is ready to classify new cases. In this paper, we ask: what do these predictions tell us about how the machine learning algorithm learns?

Our contribution is to apply methodological innovations from cognitive economics that were designed to study human cognition to instead better understand machine learning. Motivated by a long literature on perception in cognitive science, cognitive economics blends economic, psychology, and neuroscience methods to better understand the limits of human perception. For instance, cognitive imprecision and efficient coding, long central in psychology, have been shown to underlie many of the heuristics and biases that have featured prominently in behavioral economics (e.g. Woodford 2014, Enke and Graeber 2019). Sims (2003) introduced entropy-based limits to cognition in the context of sluggish and delayed responses to macroeconomic policy, and this launched a large subsequent literature on rational inattention that highlights the costs of cognition.

Our approach is to model machine learners exactly how human learners are often modeled in economics, psychology, and neuroscience: as individual decision makers who engage in signal gathering, belief formation, and choice. Initially, we follow a standard approach in cognitive economics by assuming that a machine learner engages in a two-stage decision process: first, it optimally chooses a signal structure from a feasible set of possibilities, and second, it chooses predictions that minimize expected losses given its posterior beliefs. In this model, which we call *feasibility-based machine learning*, the algorithm chooses among a feasible set of ways to learn based on the loss function it is provided.

If a machine learning algorithm is a feasibility-based learner, then it is sufficient for the data scientist to provide the machine learning algorithm with the desired objective (welfare-

[1] See Agrawal, Gans, and Goldfarb 2022 for an in-depth discussion of AI capabilities and its impact on society and business.

aware or otherwise) as its loss function, and it will learn optimally given this objective. For example, consider CheXNeXt, an influential deep learning convolutional neural network for predicting thoracic diseases from chest X-ray images (Rajpurkar, Irvin, Ball, et al. 2018). There are two factors that appear to have informed the choice of loss function in CheXNeXt. First, cases of pneumonia are rare, and it is important to prevent the algorithm from cautiously classifying cases as free of pneumonia given that this will be correct in the vast majority of cases. The second and related issue is that false negatives, when the algorithm fails to identify a case of pneumonia, are far more costly. The end result is that the algorithm is trained to minimize a function that set losses for failing to correctly identify pneumonia two orders of magnitude higher than the converse mistake.

On the surface, this seems like a reasonable procedure. If a false negative is 100 times more costly than a false positive, then it seems reasonable that the loss function on which the algorithm is trained should specify that. This logic is entirely analogous to the economic model of physical production that is based on drawing a production possibility frontier. In the economic setting, when relative prices of inputs change it is optimal to move to a corresponding point on this frontier where relative factor productivities match relative prices. Likewise if a change in the loss function for training an algorithm moves the machine around a fixed technology-based frontier, the optimal loss function must be one that reflects the relative cost of the different types of mistakes that are possible.

The problem with this logic is that it is built on a shaky premise. We show that the quality of the information that this algorithm provides degrades when the weight on misclassifying pneumonia cases is set too high. In fact, it is better for almost any reasonable objective function to use weights that are closer to equal because the machine learns more and produces information that is more useful as a result. If the optimal loss function for training has nothing to do with the relative cost of different mistakes, how then can one work out the right loss function to use to achieve an objective? This is especially important when we want the loss function to achieve certain social aims, such as in welfare-aware machine learning (Rolf et al. 2020).

We identify a path forward by translating ideas from the costly learning branch of cog-

nitive economics into our conceptualization of machine learning. Specifically, we need to think about what stops the machine from learning more than it does. For a human learner, economic theory suggests an obvious first-order answer. Learning stops when the balance of costs and benefits becomes unfavorable. As a result, the costs of learning enter the picture front and center. Following this line, we propose an alternative model of machine learning in the rational inattention tradition. In this model, which we call *cost-based machine learning*, the algorithm behaves *as if* it adjusts its learning in response to some additional unobservable costs, which we call its *pseudo-costs* because they can only be recovered empirically and may not have any relationship with any real resource inputs.[2]

Why might an algorithm act as if there are different costs to different ways of learning? In practice, this happens because the programmer of the algorithm adds code that either directly or indirectly uses the loss function to determine the extent of learning. For instance, the amount of learning could be driven by the algorithm's choice of a particular hyperparameter, which can be influenced by the loss function through "hyperparameter optimization" (e.g., Bergstra and Bengio 2012). In addition, an algorithm might perform more or fewer epochs if the losses under a particular loss function have not converged sufficiently (e.g., as implemented by the "ReduceLROnPlateau" function in Keras).

Of course, while learning may vary with the loss function, it is not immediately clear that an algorithm can be modeled as optimally balancing losses and learning costs. Somewhat remarkably, we find that changes in the loss function used when training ChexNeXt impact the nature of learning just as they might if the algorithm was a rational human being who found learning costly. In other words, the algorithm does indeed act as if it decides the stopping rule based on a pseudo-cost function. In addition, we recover sharp bounds on the structural cost parameters and construct "representative" pseudo-costs for the algorithm. These bounds and representative costs suggest that by placing relatively lower weight on pneumonia instances the algorithm acts as if it is worth incurring higher learning costs, which is sensible given that doing so places relatively higher weight on non-pneumonia instances, which are far more common.

[2]Feasibility-based learning is a special case of cost-based learning because the feasible set can be represented by a cost function that is zero for feasible ways to learn, and infinite otherwise.

Our approach could be leveraged to determine more effective loss functions for training the model given a third party's objective, such as the objective of a firm or policy maker for a particular initiative or project. That is, given their objective we could determine which loss function would result in the optimal predictions from the their perspective. For instance, even if a third party values type II errors (missing pneumonia) 99 times more than type I errors (incorrectly diagnosing pneumonia), it may nevertheless be better for the loss function to value type I and type II errors equally during training. The map between an outside party's objective and the optimal loss function critically depends on the algorithm's pseudo-costs of learning.

Further, we could ask if the revealed pseudo-learning costs align with actual costs of running the algorithm — that is, whether there is a correlation between implied costs of the algorithm and external measures of cost: run time, energy use, or epochs implemented. This would help firms to understand if the current programming makes the algorithm's pseudo-costs well-calibrated to the real costs they face.

To help in both exercises, we could specialize it to specific functional forms of revealed algorithmic pseudo-cost. For example, using Shannon entropy as the algorithm's pseudo-cost would offer very sharp predictions about the predictions of the trained model under specific loss function used in training. Sharper predictions could aid in "loss function design" by more precisely identifying the optimal loss function given an third party's objective. Even specialization to general classes of revealed algorithmic pseudo-cost, such as posterior separability, might offer substantially sharper predictions.

Our approach provides a natural join between three growing literatures: machine learning, cognitive economics, and rational inattention. Our feasibility-based machine learning model is an application of the capacity constrained learning model characterized by Caplin, Martin, and Marx (2023), which generalizes the fixed capacity version of rational inattention theory proposed by Sims (2003) and the noisy cognition model proposed by Woodford (2014). Our cost-based machine learning model is an application of the general version of rational inattention theory characterized by Caplin and Dean (2015), which itself generalizes the specialized version using Shannon entropy characterized by Matejka and McKay

(2015) and Caplin, Dean, and Leahy (2017). Thus, our paper is connected to both a growing literature that considers stochastic choice to be essential for studying limited attention in human decision making (e.g., Manzini and Mariotti 2014; Cattaneo, Ma, Masatlioglu, and Suleymanov 2020) and a growing literature that studies the theoretical properties of costly learning (e.g., Gentzkow and Kamenica 2014; De Oliveira, Denti, Mihm, and Ozbek 2017; Hébert and Woodford 2017; Denti 2022; Lipnowski and Ravid 2022). As with many of the models developed in the literature on costly learning, our models build off the core objects of information design (e.g., Kamenica and Gentzkow 2011; Bergemann and Morris 2019 ).[3]

Our analysis illustrates several advantages to applying cognitive economic methods and rational inattention theory to machine learners. First, the machine's loss function is a known and manipulable primitive of the decision problem, whereas a human's utility function must be inferred and is only indirectly manipulable. In direct contrast, Pattanayak and Krishnamurthy (2021) assume that each algorithm has an unobservable "utility" function that dictates the priorities it assigns to correct and incorrect predictions rather than treating the algorithm's objective as known and subject to external control as we do. A second advantage is that machines naturally generate state-dependent stochastic choice data (Caplin and Martin 2015), which is particularly well-suited for analyzing such models. For human decision making, such data is harder to come by.[4] Finally, machines may better approximate and emulate the costly learning paradigm. Human decisions contain strong and possibly immutable deviations from Bayesian updating and optimal choice, as documented by an extensive literature in behavioral economics. For example, a human may update beliefs in a biased manner, possibly for self-protective reasons.[5]

Despite these advantages, it is striking how little is known about algorithms as deci-

[3]Liang, Lu, and Mu (2022) draw a point of connection between information design and machine learning to study the tradeoffs between accuracy and fairness.

[4]Exceptions include sports (Archsmith, Heyes, Neidell, and Sampat 2021; Bhattacharya and Howard 2021; Almog, Gauriot, Page, and Martin 2024), quality control settings, and lab experiments (Dean and Neligh 2017; Almog and Martin 2023). When observational data falls short because of issues such as selective labels, this requires further econometric work to address (Rambachan 2021).

[5]However, future work may uncover that machine learning algorithms can be modeled as having their own important set of biases that lead to departures from standard models of feasibility-based and cost-based learning.

sion makers.[6] We demonstrate that cognitive economic methods and rational inattention theory can be applied to machine learners using standard machine learning data sets and optimization procedures. We hope that this opens the door to better understanding these increasingly important learners.

The rest of the paper proceeds as follows. Section 2 introduces the preliminaries of our model and empirical application. Section 3 covers the foundation for our learning models and looks for consistency with this foundation in our empirical application. Building on this foundation, Section 4 covers the feasibility- and cost-based machine learning models, our approach to cost recovery, and applies them in our empirical setting.

# 2 Preliminaries

## 2.1 Setup and Notation

There is a finite set of *outcomes* $Y$ (e.g., image types, disease severity levels, etc.) that the algorithm can learn about. There is also a set of *predictions* $A$ that the algorithm can make. For example, many classification algorithms output a numeric *confidence score* for each possible outcome. In turn, confidence scores can be translated into discrete outcome predictions using a downstream classification rule, such as predicting an outcome if its confidence score exceeds a given threshold or the confidence scores of the other possible outcomes.

In our running application, the set of possible outcomes $Y = \{0, 1\}$ is an indicator for the presence of pneumonia, and the set of possible predictions $A = [0, 1]$ is a continuous measure of the algorithm's "confidence" about the presence of pneumonia. In our application, we consider numeric confidence scores instead of discrete downstream outcome predictions because confidence scores are more closely tied to machine incentives and yield stronger tests and sharper identification. Our framework also accommodates situations where the analyst only has data on predicted outcomes or wishes to model the scoring algorithm jointly with

[6] Notable exceptions include Zhao, Ke, Wang, and Hsieh (2020), who embed behavioral forces in a neural net structure, and Danan, Gajdos, and Tallon (2020), who apply decision-theoretic approaches to recommendation systems.

a downstream classification rule.

An important input to the training of an algorithm is a *loss function* $L : A \times Y \to \mathbb{R}$, which indicates the value of a particular prediction given the outcome. It is standard practice to use cross entropy $L(a, y) = -y \log a - (1-y) \log(1-a)$ over confidence scores and outcomes when training deep learning neural networks to predict the class to which an observation belongs. In addition, it is also standard practice to re-weight the loss function to make losses higher or lower for a particular outcome; such class weighting is often employed when one outcome is less common (Thai-Nghe, Gantner, and Schmidt-Thieme 2010) or with the hope of achieving some external objective (Zadrozny, Langford, and Abe 2003).

For a given algorithm, using loss function $L$ in training generates a series of predictions $a_1, ..., a_N$, one for each observation in a test data set of size $N$.[7] The performance of the trained model is assessed on how well its predictions align with actual outcomes. To perform this assessment, the analyst has access to the actual outcomes for observations in the test data, which is a series of outcomes $y_1, ..., y_N$.[8] We follow the standard practice of evaluating how well an algorithm performs on aggregate for each outcome. Formally, aggregate level performance for each outcome is summarized by *performance data* $P^L : A \times Y \to [0, 1]$, which is the joint distribution of predictions and outcomes for the trained model in a test data set,

$$P^L(a, y) = \frac{1}{N} \sum_{n \in \{1,...,N\}} \mathbf{1}_{a_n = a \,\&\, y_n = y}$$

Because the test data set is finite, the support of $P^L$ over $A$, given by $\text{supp}(P^L_A)$, is also finite. As is standard practice in the machine learning literature, we study algorithmic predictions over a test sample that is independently drawn from the same population as the data on which the algorithm is trained.

[7] Our approach could also be used to study the algorithm's predictions in the training data set.

[8] Specifically, Wang et al. (2017) actual outcomes by labeling whether each chest X-ray indicates pneumonia or not. As is standard when evaluating algorithms, we assume this *ground truth* is correct, but our approach could be extended to include uncertainty about the ground truth.

## 2.2 Application: Data and Algorithm

We demonstrate the potential empirical suitability of our machine learning models by revisiting CheXNeXt, an influential deep convolutional neural network for predicting thoracic diseases from chest X-ray images (Rajpurkar, Irvin, Ball, et al. 2018). Our training models are generated using the ChestX-ray14 data set, which consists of 112,120 frontal chest X-rays which were synthetically labeled with the presence of fourteen thoracic diseases (Wang et al. 2017). The main modifications we make to the CheXNeXt training procedure are that we isolate the task of pneumonia detection as in the earlier implementation of Rajpurkar, Irvin, Zhu, et al. (2017), and we train the algorithm across various $\beta$-weighted cross entropy loss functions:

$$L^{\beta}(a, y) = -\beta y \log(a) - (1 - \beta)(1 - y) \log(1 - a). \tag{1}$$

Specifically, we vary the loss function by considering $\beta = 0.7, 0.9, 0.99$.[9] In addition, we employ ensemble (model-averaging) methods to isolate the substantive effects of what the machine learns from random noise inherent to the stochastic training procedure. Using nested cross-validation methods, this yields an ensemble model prediction at each $\beta$ for each of the 112,120 X-ray images in the original data. Further technical details of our training procedure are relegated to Appendix A. We return to our application in Subsection 3.2 after introducing our fundamental representation of machines as Bayesian expected loss minimizers.

# 3 Machines as Bayesian Expected Loss Minimizers

In this section we present the cognitive economic foundation of the learning models we consider, the testable implications of this foundation, and positive evidence of this foundation in our empirical application.

In this foundation we follow the standard perceptual model of signal processing and choice. For both feasibility-based and cost-based machine learning, we model an algorithm as an optimizing agent that i) starts with a prior $\mu \in \Delta(Y)$ over outcomes, ii) gets signal

[9] For reference, the class weight used in the analysis of Rajpurkar, Irvin, Zhu, et al. (2017) is approximately 0.99 because the probability of positive pneumonia cases in the data set is 0.0127.

realizations that provide information about the outcome, iii) forms posterior beliefs $\gamma \in \Delta(Y)$ via Bayesian updating, and iv) chooses predictions based on these posteriors to minimize expected losses. As in Kamenica and Gentzkow (2011) we define $\mathcal{Q}$ as those distributions of posteriors with finite support that satisfy Bayes' rule,

$$\mathcal{Q} \equiv \{Q \in \Delta(\Delta(Y)) | \sum_{\gamma \in \text{supp}(Q)} \gamma Q(\gamma) = \mu\}.$$

Posteriors are translated into probabilistic predictions through a prediction function $q : \text{supp}(Q) \to \Delta(A)$. For a given loss function $L$ and distribution of posteriors $Q \in \mathcal{Q}$, the set of optimal prediction functions is defined as,

$$\hat{q}(L, Q) \equiv \underset{q}{\text{argmin}} \sum_{\gamma \in \text{supp}(Q)} Q(\gamma) \sum_{a \in A} q(a|\gamma) \sum_{y \in Y} \gamma(y) L(a, y).$$

Note that any pair $(Q, q)$ produces a joint distribution of predictions and outcomes given by $P_{(Q,q)} : A \times Y \to [0, 1]$ where,

$$P_{(Q,q)}(a, y) \equiv \sum_{\gamma \in \text{supp}(Q)} Q(\gamma) q(a|\gamma) \gamma(y).$$

With these elements in place we can define the foundation of our subsequent learning models.

**Definition 1.** *For a given loss function $L$, $P^L$ has a **signal-based representation** (SBR) if there exists a prior $\mu \in \Delta(Y)$, a Bayes consistent distribution of posteriors $Q \in \mathcal{Q}$, and a prediction function $q : \text{supp}(Q) \to \Delta(A)$ such that:*

1. *The prior is correct: $\mu(y) = \sum_{a \in \text{supp}(P^L_A)} P^L(a, y)$.*
2. *Predictions are optimal at all possible posteriors: $q \in \hat{q}(L, Q)$.*
3. *Predictions are generated by the model: $P^L(a, y) = P_{(Q,q)}(a, y)$.*

If $P^L$ has an SBR, then it is *as if* the algorithm makes predictions to minimize the loss function given the Bayesian posterior beliefs induced by its signal structure.

## 3.1 Testable Condition: Loss Calibration

Applying the theoretical results of Caplin and Martin (2015) and Bergemann and Morris (2016), it is straightforward to show that the SBR foundation is characterized by a simple condition, called *loss calibration*, which requires that switching wholesale from any prediction $a$ to any alternative prediction $a'$ would never strictly reduce losses.

**Definition 2.** *Performance data $P^L$ is **loss calibrated** to loss function $L$ if a wholesale switch of predictions does not reduce losses according to $L$:*

$$a \in \underset{a' \in A}{\operatorname{argmin}} \sum_{y \in Y} P^L(a, y) L(a', y) \text{ for all } a \in \operatorname{supp}(P^L_A).$$

Any algorithm that fails this condition makes predictions that are not suitable for the loss function and that are thus inconsistent with an SBR and Bayesian expected loss minimization.[10]

Under weighted cross entropy loss with binary outcomes (1), it is straightforward to show that loss calibration takes a unique closed form as a function of posterior probabilities. Let $a^\beta(\gamma)$ be the optimal prediction when the class weight is $\beta$ and the posterior probability that the outcome is $y = 1$ is given by $\gamma$.

**Observation 1.** *Consider weighted cross entropy loss (1). For any weight $\beta \in (0, 1)$ and all posterior probabilities $\gamma \in \operatorname{supp}(Q)$ that the outcome is $y = 1$, the unique loss calibrated confidence score is given by:*

$$a^\beta(\gamma) = \frac{\beta\gamma}{1 - \beta - \gamma + 2\beta\gamma}. \tag{2}$$

In the case of unweighted cross entropy loss ($\beta = 0.5$), the loss calibrated scoring function (2) collapses to the optimal prediction being the posterior probability itself, $\alpha^{0.5}(\gamma) = \gamma$. Thus, as is well-known, unweighted cross entropy incentivizes truthful revelation of beliefs.

[10] Nevertheless, this is easy to rectify. Whenever this loss function is input into the algorithm, a single line of code at the end of the computer program making a wholesale switch to predicting $a$ whenever it would have predicted $a'$ would make this algorithm loss calibrated for this loss function.

## 3.2 Application: Loss Calibration

In the deep learning algorithm we consider — which regularizes through early stopping and aggregates over an ensemble of trained neural nets — we find that confidence scores are loss calibrated as in (2) across various weights in weighted cross entropy loss (1). Recall that this includes unconditional calibration for unweighted cross entropy loss ($\beta = 0.5$), which is a proper loss function.

Graphical evidence of calibration and loss calibration is provided in the left and right panels of Figure 1, respectively. In each plot, the horizontal axis represents the confidence score, and the vertical axis the corresponding pneumonia rate in the data (both on a log scale). The shapes in the figure provide the empirical decile-binned calibration curves (DeGroot and Fienberg 1983; Niculescu-Mizil and Caruana 2005). The solid lines represent the situation where confidence scores are calibrated. Thus, the algorithm appears effectively calibrated for the unweighted loss function $\beta = 0.5$, and miscalibrated otherwise. The dashed lines in the right plot show the theoretical relationship between scores and pneumonia rates for the relative positive class weights $\beta = 0.7, 0.9, 0.99$ if an algorithm is loss calibrated (note that the loss calibrated and calibrated lines coincide on the left when $\beta = 0.5$). As $\beta$ increases, the algorithm is increasingly incentivized to provide a score that is higher than the machine's actual "belief" about the probability of pneumonia, which causes the lines to bow out. The alignment of theoretical predictions and empirical estimates strongly suggests that the algorithm is generally very close to being loss calibrated, and very close to being calibrated when the loss function is unweighted.

Finally, note that our finding of calibration with an unweighted loss function is consistent with previously documented calibration for deep learning convolutional neural networks that use regularization (i.e., weight decay in Guo, Pleiss, Sun, and Weinberger 2017) and deep ensembling (Lakshminarayanan, Pritzel, and Blundell 2017). With a viable SBR in hand, we now turn to our models of what the machine learns.

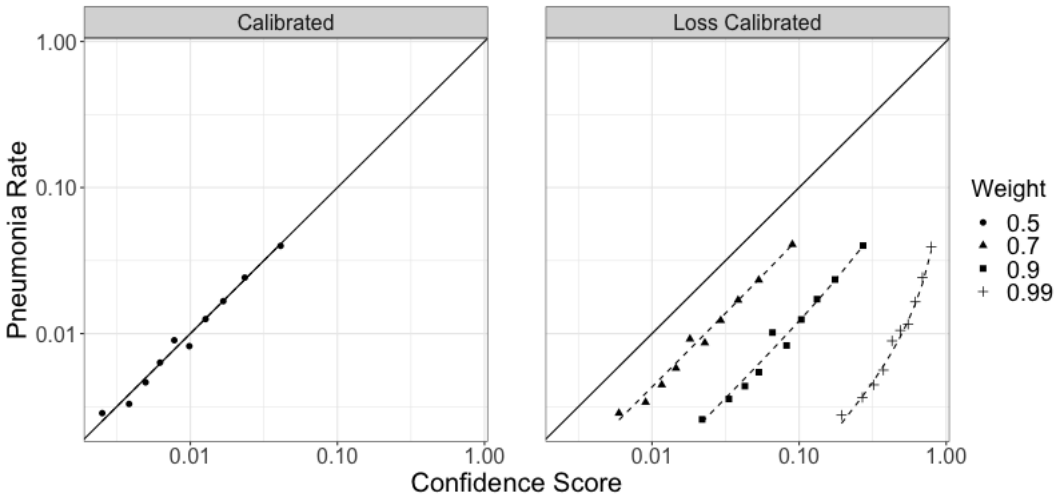


Figure 1: Theoretical relationship between confidence score and pneumonia rate for a loss calibrated algorithm with calibration target (solid lines), loss calibration targets varying class weights (dashed lines), and empirical decile-binned calibration curves (shapes) for the pneumonia-detection algorithm presented in Subsection 2.2. All objects are displayed on a log-10 scale to improve readability. This figure provides visual evidence that the algorithm is calibrated for $\beta = 0.5$ (left panel) and loss calibrated generally (left and right panels). This ensures an SBR representation and simplifies computation of the objects introduced in Section 4.

# 4 Models of Machine Learning

SBR leaves open the question of how a machine learning algorithm arrives at its signal structure – that is, what the machine decides to learn based on the incentives provided by the loss function. We now propose and characterize two nested alternatives: choosing among a set of feasible signal structures or choosing among signal structures of different costs.

## 4.1 Feasibility-Based Machine Learning

Our first model assumes the algorithm chooses among a set of feasible signal structures to best match the incentives provided by the loss function.

To translate this into the SBR framework of Section 3, we define a feasible set of experiments $\mathcal{Q}^* \subset \mathcal{Q}$. This feasible set depends only on the algorithm's capability and is not specific to the loss function provided. We define the algorithm's strategy space $\Lambda$ to include both $Q$ and $q$:

$$\Lambda = \{(Q, q) | Q \in \mathcal{Q}, q : \text{supp}(Q) \to \Delta(A)\}.$$

For a given loss function $L$ and feasible set $\mathcal{Q}^*$, the set of optimal strategies $\tilde{\Lambda}(L, \mathcal{Q}^*)$ is,

$$\tilde{\Lambda}(L, \mathcal{Q}^*) \equiv \underset{(Q,q)\in\Lambda, Q\in\mathcal{Q}^*}{\text{argmin}} \sum_{\gamma\in\text{supp}(Q)} Q(\gamma) \sum_{a\in A} q(a|\gamma) \sum_{y\in Y} \gamma(y) L(a, y).$$

With this we can define all performance data sets that are consistent with optimality for a given feasible set $\mathcal{Q}^*$ as,

$$\tilde{P}(L, \mathcal{Q}^*) \equiv \{P_{(Q,q)} | (Q, q) \in \tilde{\Lambda}(L, \mathcal{Q}^*)\}.$$

Feasibility-based machine learning requires that there exist a feasible set $\mathcal{Q}^*$ such that the performance data produced by an algorithm are optimal given that feasible set for all $L \in \mathcal{L}$.

**Definition 3.** *An algorithm is consistent with* ***feasibility-based machine learning*** *if*

*there exists a feasible set* $\mathcal{Q}^* \subset \mathcal{Q}$ *such that* $P^L \in \tilde{P}(L, \mathcal{Q}^*)$ *for all* $L \in \mathcal{L}$.

## 4.2 Cost-Based Machine Learning

Our second model assumes the algorithm chooses among signal structures of different costs. Once again, these are not the monetary costs incurred in running the algorithm, but rather reflect the intrinsic difficulty the algorithm has in learning the true outcome given its training procedures and the available training data. One way to interpret these costs is as the resource costs of different mathematical operations.

To formalize this, we define a learning cost function $K : \mathcal{Q} \to \mathbb{R}$ and denote the set of all possible learning cost functions as $\mathcal{K}$. An algorithm's learning cost function depends only on its capabilities and is not specific to the loss function provided. Given loss function $L \in \mathcal{L}$ and learning cost function $K \in \mathcal{K}$, the *resource-adjusted loss* $\hat{L}$ of strategy $(Q, q)$ is,

$$\hat{L}((Q,q)|L,K) \equiv \sum_{\gamma \in \text{supp}(Q)} Q(\gamma) \sum_{a \in A} q(a|\gamma) \sum_{y \in Y} \gamma(y) L(a,y) + K(Q).$$

The corresponding set of optimal strategies $\hat{\Lambda}(L, K)$ is then defined as,

$$\hat{\Lambda}(L,K) \equiv \underset{(Q,q)\in\Lambda}{\text{argmin}} \ \hat{L}((Q,q)|L,K).$$

This optimization problem formalizes the way in which the algorithm trades off losses with learning costs. Given any $L \in \mathcal{L}$ the set of all performance data sets that are consistent with optimality for a given learning cost function $K \in \mathcal{K}$ are,

$$\hat{P}(L,K) \equiv \{P_{(Q,q)} | (Q,q) \in \hat{\Lambda}(L,K)\}.$$

**Definition 4.** *An algorithm is consistent with* ***cost-based machine learning*** *if there exists a learning cost function* $K \in \mathcal{K}$ *such that* $P^L \in \hat{P}(L, K)$ *for all* $L \in \mathcal{L}$.

The second learning model generalizes the first model because a feasible set of posterior distributions $\mathcal{Q}^*$ is equivalently specified as an indicator learning cost function $K^*$, for which

the cost is zero for every feasible posterior distribution $Q \in \mathcal{Q}^*$ and infinite otherwise.

### 4.3 Testing the Models

In order to simplify the characterizations of these models, we restrict consideration to performance data sets with an SBR representation (or its empirically verifiable counterpart, loss calibration). Because indexing is useful in what follows, we take as given a finite set of $M$ loss functions, indexed by $1 \leq m \leq M$. For notational simplicity, we denote the performance data set from training the algorithm with the $m$-th loss function as $P^m = P^{L^m}$.

Our characterization of feasibility-based learning is taken directly from Caplin, Martin, and Marx (2023), who characterize capacity constrained learning for human decision makers. In the context of machine learning, the key idea is to ensure that losses cannot be lowered by counterfactually switching to the predictions from training with a different loss function. All such comparisons are visible in the *value of learning (VoL) matrix* $G$ with generic element $G^{mn}$ in row $m$ and column $n$ that specifies the minimized expected losses when the loss function is $L^m$ and the performance data is $P^n$:

$$G^{mn} \equiv \sum_{a \in \text{supp}(P^n_A)} \min_{a' \in A} \sum_{y \in Y} L^m(a', y) P^n(a, y).$$

The operation on the right hand side of the equation takes any prediction $a \in \text{supp}(P^n_A)$, picks some alternative prediction $a' \in A$ to replace it wholesale, computes the corresponding expected losses for $L^m$, and then minimizes.

A feasibility-based machine learning representation requires that no such switch of performance data can lower losses. To formalize we define the $M \times M$ *direct value difference matrix* $D_0$ by,

$$D_0^{mn} \equiv G^{mn} - G^{mm}. \tag{3}$$

An algorithm with an SBR is *strongly loss adapted* if for all $1 \leq m, n \leq M$,

$$D_0^{mn} \geq 0, \text{ or equivalently } G^{mn} \geq G^{mm}.$$

The results in Caplin, Martin, and Marx (2023) can be used to show that, together with loss calibration, an algorithm being strongly loss adapted is necessary and sufficient for feasibility-based machine learning. To apply their results one maps utility maximization to loss minimization and action sets to loss functions.

The corresponding characterization of cost-based learning is based on paths of switches. Define $H(m,n)$ as all sequences of indices $\vec{h} = (h(1), h(2), \ldots, h(J(\vec{h})+1))$ of edge length $J(\vec{h})$ with $h(1) = m$ and $h(J(\vec{h})+1) = n$ in which the first $J(\vec{h})$ entries are distinct. The *indirect value difference matrix* $D$ computes minimizing summed loss differences on such paths

$$D^{mn} \equiv \min_{\{\vec{h} \in H(m,n)\}} \sum_{j=1}^{J(\vec{h})} D_0^{h(j)h(j+1)}. \tag{4}$$

Formally, an algorithm with an SBR is *loss adapted* if for all $1 \leq m \leq M$,

$$D^{mm} \geq 0.$$

This condition requires that no cycle of switches could lower losses. Applying the results in Caplin and Dean (2015), an algorithm with an SBR has a cost-based explanation if and only if it is loss adapted.

Caplin, Martin, and Marx (2023) show that if an algorithm is loss adapted, then the value difference matrix can be computed in polynomial time by applying the Floyd-Warshall algorithm (Floyd 1962, Warshall 1962) to the complete weighted directed graph with weight $D_0^{mn}$ on the directed edge from node $1 \leq m \leq M$ to node $1 \leq n \leq M$. Loss adaptedness is easily verified by the Floyd-Warshall procedure: an algorithm is loss adapted if and only if the candidate matrix thus computed has a zero diagonal.

## 4.4 Recovering Costs of Learning

Building on Caplin and Dean (2015), Caplin, Martin, and Marx (2023) show how to identify all *qualifying* costs $\{K^m\}_{m=1}^M$ that rationalize the observed performance data according to cost-based learning. This is when costs are minimized by choosing performance data set $P^m$

at cost $K^m$ when the loss function is $L^m$

$$D_0^{mm} + K^m \leq D_0^{mn} + K^n, \tag{5}$$

for all $1 \leq m, n \leq M$. For present purposes, a key observation is that, normalizing to $K^M = 0$, qualifying learning costs $\{K^m\}_{m=1}^M$ define a convex polyhedron in $\mathbb{R}^{M-1}$ with the sign-inverted $M$-th row $(-D^{M1}, \cdots, -D^{M(M-1)})$ and $M$-th column $(D^{1M}, \cdots, D^{(M-1)M})$ providing a subset of the extreme points. Furthermore, the average cost $\bar{K}$ across normalizations is potentially appealing as a representative learning cost because it is "central," qualifying, and easy to compute.[11]

## 4.5 Application

The main products of our empirical application are estimates of the VoL matrix $G$ and indirect value difference matrix $D$ introduced in the previous Subsection 4.3. To facilitate their computation, we rely on the strong evidence for an SBR representation provided in Subsection 3.2. This strong evidence of loss calibration allows us to compute losses in the $G$ matrix, as given below, by analytically recalibrating confidence scores inverting (2) to recover optimal confidence scores across weights.[12]

$$G = 0.01 \begin{array}{c} \\ L^{0.7} \\ L^{0.9} \\ L^{0.99} \end{array} \begin{array}{c} \begin{array}{ccc} P^{0.7} & P^{0.9} & P^{0.99} \end{array} \\ \begin{pmatrix} 3.750 & 3.752 & 3.762 \\ 3.349 & 3.352 & 3.362 \\ 1.365 & 1.366 & 1.370 \end{pmatrix} \end{array}$$

[11] Denti (2022) provides a linear program that recovers the minimum learning cost function in which not learning is free. For human decision-makers, it is natural to assume that inattention is free, but this assumption is less natural for algorithms. Also, the minimal monotone learning cost might be quite different from the rest of the qualifying learning costs, and so might not be very representative.

[12] In turn, this analytical mapping circumvents the need to bin data to recover posterior beliefs, avoiding the finite sample issues associated therewith.

Recall that a necessary and sufficient condition for feasibility-based machine learning is that the algorithm is strongly loss adapted:

$$\mathcal{H}_0 : D_0^{mn} \equiv G^{mn} - G^{mm} \geq 0 \quad \text{for all } 1 \leq m, n \leq M.$$

In order to statistically test this multivariate one-sided hypothesis, we first estimate a $9 \times 9$ covariance matrix for the VoL matrix $G$ via 10,000 bootstrap samples from the data set of ensemble predictions. We then compute $p$-values for the constituent univariate one-sided Wald tests and apply a Bonferroni correction. Even using this conservative approach to bounding the family-wise error rate, we reject the null hypothesis at standard levels of significance with $p = 0.0014$. Further inspection of $G$ reveals a systematic reason for why we reject the null hypothesis: loss functions have a common preference for the performance data from training with lower $\beta$. Thus, while the predictions for the loss function with weight $\beta = 0.7$ are consistent with the algorithm being strongly loss adapted, the predictions for the loss functions with weight $\beta = 0.9, 0.99$ are not.

Our second question is whether the algorithm is loss adapted, and thereby consistent with cost-based learning:

$$\mathcal{H}_0 : D^{mm} \geq 0 \quad \text{for all } 1 \leq m \leq M.$$

The estimated $D$ matrix, given below, satisfies this null hypothesis.

$$D = 0.1^5 \begin{array}{c} \begin{array}{ccc} P^{0.7} & P^{0.9} & P^{0.99} \end{array} \\ \begin{pmatrix} 0 & 2.548 & 12.467 \\ -2.529 & 0 & 9.938 \\ -6.993 & -4.463 & 0 \end{pmatrix} \end{array} \begin{array}{c} L^{0.7} \\ L^{0.9} \\ L^{0.99} \end{array}$$

We therefore fail to reject that the algorithm is consistent with cost-based learning at any level of significance.[13] Intuitively, a necessary condition for this is that, even though all loss functions are minimized by switching to lower-$\beta$ performance data, the gains from switching

[13] Pointwise consistency with loss adaptedness is satisfied in 53% of our bootstrap samples.

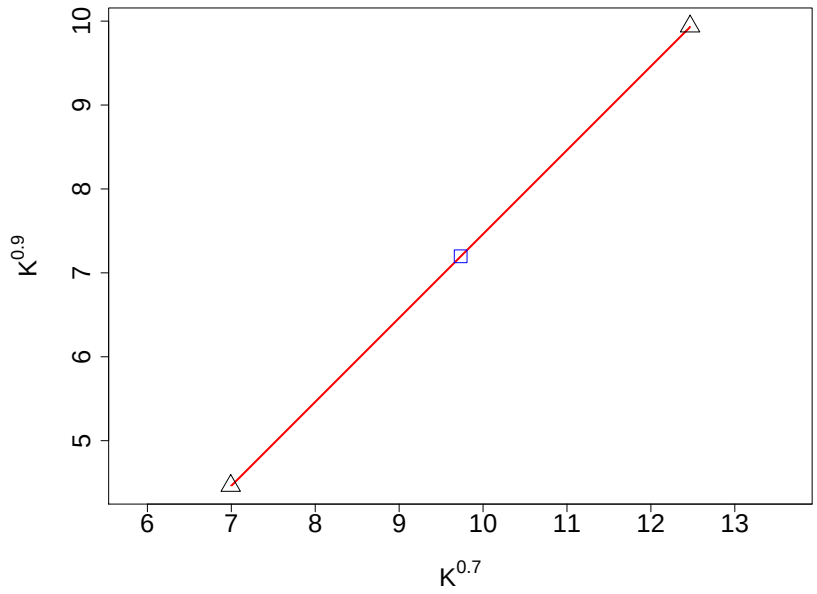


Figure 2: Recovery of cost polyhedron (red) when $K^{0.99}$ is normalized to zero, with two extreme points (triangles) and the representative cost (square) marked. The relationship between $K^{0.7}$ and $K^{0.9}$ is nearly point identified by the data. Furthermore, any rationalizing cost function has the feature that $K^{0.99} \leq K^{0.9} \leq K^{0.7}$. For legibility, the axis values are re-scaled by a factor of $10^5$.

are lower for higher-$\beta$ loss functions.

Because the algorithm in our application is consistent with cost-based learning as in Section 4, we can also recover all qualifying costs. Normalizing $K^{0.99} = 0$, the cost polyhedron for remaining costs $K^{0.7}$ and $K^{0.9}$ is illustrated in Figure 2. The figure also plots the pair of extreme costs identified by $D$ (triangles) and the representative cost (square). Given that the value of performance data is decreasing in the $\beta$ of the weighted loss function, the costs of learning must also be decreasing in $\beta$ in order to rationalize the observed choices.

Finally, we conclude with a brief observation on the power of our test for cost-based learning: *any* reordering of chosen information structures would have resulted in a pointwise rejection of loss adaptedness. Thus, we failed to reject the null hypothesis in spite of — rather than in the absence of — a powerful test.

# References


Agrawal, Ajay, Joshua Gans, and Avi Goldfarb (2022). *Prediction Machines, Updated and Expanded: The Simple Economics of Artificial Intelligence*. Harvard Business Press.

Almog, David, Romain Gauriot, Lionel Page, and Daniel Martin (2024). "AI Oversight and Human Mistakes: Evidence from Centre Court". In: *Working Paper*.

Almog, David and Daniel Martin (2023). "Rational Inattention in Games: Experimental Evidence". In: *Working Paper*.

Archsmith, James E, Anthony Heyes, Matthew J Neidell, and Bhaven N Sampat (2021). *The Dynamics of Inattention in the (Baseball) Field*. Tech. rep. National Bureau of Economic Research.

Bai, Yu, Song Mei, Huan Wang, and Caiming Xiong (2021). "Don't just blame over-parameterization for over-confidence: theoretical analysis of calibration in binary classification". In: *arXiv preprint arXiv:2102.07856*.

Bergemann, Dirk and Stephen Morris (2016). "Bayes correlated equilibrium and the comparison of information structures in games". In: *Theoretical Economics* 11.2, pp. 487–522.

— (2019). "Information design: A unified perspective". In: *Journal of Economic Literature* 57.1, pp. 44–95.

Bergstra, James and Yoshua Bengio (2012). "Random search for hyper-parameter optimization." In: *Journal of machine learning research* 13.2.

Bhattacharya, Vivek and Greg Howard (2021). "Rational Inattention in the Infield". In: *American Economic Journal: Microeconomics*.

Caplin, Andrew and Mark Dean (2015). "Revealed preference, rational inattention, and costly information acquisition". In: *American Economic Review* 105.7, pp. 2183–2203.

Caplin, Andrew, Mark Dean, and John Leahy (2017). *Rationally inattentive behavior: Characterizing and generalizing Shannon entropy*. Tech. rep. National Bureau of Economic Research.

Caplin, Andrew and Daniel Martin (2015). "A testable theory of imperfect perception". In: *The Economic Journal* 125.582, pp. 184–202.

Caplin, Andrew, Daniel Martin, and Philip Marx (2023). "Rationalizable Learning". In: *NBER Working Paper series*.

Cattaneo, Matias D, Xinwei Ma, Yusufcan Masatlioglu, and Elchin Suleymanov (2020). "A random attention model". In: *Journal of Political Economy* 128.7, pp. 2796–2836.

Danan, Eric, Thibault Gajdos, and Jean-Marc Tallon (2020). "Tailored recommendations". In: *Social Choice and Welfare*, pp. 1–20.

De Oliveira, Henrique, Tommaso Denti, Maximilian Mihm, and Kemal Ozbek (2017). "Rationally inattentive preferences and hidden information costs". In: *Theoretical Economics* 12.2, pp. 621–654.

Dean, Mark and Nate Leigh Neligh (2017). "Experimental tests of rational inattention". In.

DeGroot, Morris H and Stephen E Fienberg (1983). "The comparison and evaluation of forecasters". In: *Journal of the Royal Statistical Society: Series D (The Statistician)* 32.1-2, pp. 12–22.

Deng, Jia et al. (2009). "Imagenet: a large-scale hierarchial image database". In: *Computer Vision and Pattern Recognition (CVPR)*, pp. 248–255.

Denti, Tommaso (2022). "Posterior separable cost of information". In: *American Economic Review* 112.10, pp. 3215–59.

Enke, Benjamin and Thomas Graeber (2019). *Cognitive uncertainty*. Tech. rep. National Bureau of Economic Research.

Floyd, Robert W (1962). "Algorithm 97: shortest path". In: *Communications of the ACM* 5.6, p. 345.

Gentzkow, Matthew and Emir Kamenica (2014). "Costly persuasion". In: *American Economic Review* 104.5, pp. 457–62.

Guo, Chuan, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger (2017). "On calibration of modern neural networks". In: *International Conference on Machine Learning*. PMLR, pp. 1321–1330.

Hébert, Benjamin and Michael Woodford (2017). *Rational inattention and sequential information sampling*. Tech. rep. National Bureau of Economic Research.

Huang, Gao, Zhuang Liu, Kilian Q. Weinberger, and Laurens van der Maaten (2016). "Densely connected convolutional networks". In: *arXiv preprint arXiv:1608.06993*.

Ioffe, Sergey and Christian Szegedy (2015). "Batch normalization: accelerating deep network training by reducing internal covariate shift". In: *International Conference on Machine Learning (ICML)*, pp. 448–456.

Kamenica, Emir and Matthew Gentzkow (2011). "Bayesian persuasion". In: *American Economic Review* 101.6, pp. 2590–2615.

Kingma, Diederik and Jimmy Ba (2014). "Adam: a method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980*.

Lakshminarayanan, Balaji, Alexender Pritzel, and Charles Blundell (2017). "Simple and scalable predictive uncertainty estimation using deep ensembles". In: *Advances in neural information processing systems 31*.

Liang, Annie, Jay Lu, and Xiaosheng Mu (2022). "Algorithmic design: Fairness versus accuracy". In: *Proceedings of the 23rd ACM Conference on Economics and Computation*, pp. 58–59.

Lipnowski, Elliot and Doron Ravid (2022). "Predicting Choice from Information Costs". In: *arXiv preprint arXiv:2205.10434*.

Liu, Sheng et al. (2022). "Deep probability estimation". In: *arXiv preprint arXiv:2111.10734*.

Manzini, Paola and Marco Mariotti (2014). "Stochastic choice and consideration sets". In: *Econometrica* 82.3, pp. 1153–1176.

Matejka, Filip and Alisdair McKay (2015). "Rational inattention to discrete choices: A new foundation for the multinomial logit model". In: *American Economic Review* 105.1, pp. 272–98.

Niculescu-Mizil, Alexandru and Rich Caruana (2005). "Predicting good probabilities with supervised learning". In: *Proceedings of the 22nd international conference on Machine learning*, pp. 625–632.

Pattanayak, Kunal and Vikram Krishnamurthy (2021). "Behavioral Economics Approach to Interpretable Deep Image Classification. Rationally Inattentive Utility Maximization Explains Deep Image Classification". In: *arXiv preprint arXiv:2102.04594*.

Rajpurkar, Pranav, Jeremy Irvin, Robyn L Ball, et al. (2018). "Deep learning for chest radiograph diagnosis: A retrospective comparison of the CheXNeXt algorithm to practicing radiologists". In: *PLoS medicine* 15.11, e1002686.

Rajpurkar, Pranav, Jeremy Irvin, Kaylie Zhu, et al. (2017). "CheXNet: radiologist-level pneumonia detection on chest x-rays with deep learning". In: *arXiv preprint arXiv:1711.05225*.

Rambachan, Ashesh (2021). *Identifying prediction mistakes in observational data*.

Rolf, Esther et al. (2020). "Balancing competing objectives with noisy data: Score-based classifiers for welfare-aware machine learning". In: *International Conference on Machine Learning*. PMLR, pp. 8158–8168.

Sims, Christopher A (2003). "Implications of Rational Inattention". In: *Journal of Monetary Economics* 50.3, pp. 665–690.

Thai-Nghe, Nguyen, Zeno Gantner, and Lars Schmidt-Thieme (2010). "Cost-sensitive learning methods for imbalanced data". In: *The 2010 International joint conference on neural networks (IJCNN)*. IEEE, pp. 1–8.

Wang, Xiaosong et al. (2017). In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2097–2106.

Warshall, Stephen (1962). "A theorem on boolean matrices". In: *Journal of the ACM (JACM)* 9.1, pp. 11–12.

Woodford, Michael (2014). "Stochastic choice: An optimizing neuroeconomic model". In: *American Economic Review* 104.5, pp. 495–500.

Zadrozny, Bianca, John Langford, and Naoki Abe (2003). "Cost-sensitive learning by cost-proportionate example weighting". In: *Third IEEE international conference on data mining*. IEEE, pp. 435–442.

Zhao, Chen, Shaowei Ke, Zhaoran Wang, and Sung-Lin Hsieh (2020). "Behavioral Neural Networks". In: *Available at SSRN 3633548*.

# A Application: Technical Details

Here we summarize the technical details of Section 2.2. Our model training procedure essentially follows that of the CheXNeXt algorithm (Rajpurkar, Irvin, Ball, et al. 2018), in which a deep neural network was trained using the ChestX-ray14 data set of Wang et al. (2017). The ChestX-ray14 data set consists of 112,120 frontal chest X-rays that were synthetically labeled with up to fourteen thoracic diseases. Our code for model training is adapted from the publicly available CheXNeXt codebase of Rajpurkar, Irvin, Ball, et al. (2018). However, we follow the earlier CheXNet implementation of Rajpurkar, Irvin, Zhu, et al. (2017) in three ways. First, we restrict to the binary classification task of pneumonia detection, where the labels of interest are pneumonia ($y = 1$) or not ($y = 0$). In addition, we trade off a higher batch rate of 16 at the expense of a slightly smaller imaging scaling size of 224 by 224 pixels (instead of a batch size of 8 and an image rescaling of 512 by 512 pixels, respectively). As in Rajpurkar, Irvin, Ball, et al. (2018), we adopt random horizontal flipping, and normalize based on the mean and standard deviation of images in the ImageNet data set (Deng et al. 2009). For each model, we train a 121-layer dense convolutional neural network (DenseNet, Huang, Liu, Weinberger, and Maaten 2016) with network weights initialized to those pretrained on ImageNet, using Adam with standard parameters 0.9 and 0.999 (Kingma and Ba 2014), and using batch normalization (Ioffe and Szegedy 2015). We use an initial learning rate of 0.0001 that is decayed by a factor of 10 each time the validation loss plateaus after an epoch, and we conduct early stopping based on validation loss. Each model was trained using either an Nvidia Tesla V100 16GB GPU or an Nvidia Tesla A100 40GB GPU on university high performance computing clusters.

Given the inferential nature of our exercise, we deviate substantively from this prior art in two ways. First, we induce variation in the cross entropy loss function (1) across multiple positive class weights $\beta_1 = 0.7, 0.9, 0.99$, with 0.99 approximately equal to the inverse probability class weights for pneumonia detection adopted in Rajpurkar, Irvin, Zhu, et al. (2017). In addition to varying class weights, the main difference in our implementation and the implementation of Rajpurkar, Irvin, Zhu, et al. (2017) are our data splits and our recourse to additional ensemble methods to account for randomness in the training

procedure. This use of ensemble methods also likely explains why our confidence scores are loss calibrated, despite recent evidence that deep neural networks and cross entropy loss may inherently produce poor calibration because of overconfidence (Bai, Mei, Wang, and Xiong 2021, Liu et al. 2022). Specifically, we adopt a nested cross-validation approach where we randomly split the data set into ten approximately equal folds and then iterate through 70-20-10 train-validation-test splits (the split distribution also used in Wang et al. 2017 and a secondary application of Rajpurkar, Irvin, Zhu, et al. 2017). We train a total of 480 models, yielding an ensemble of 96 trained models for each observation in the data set where that observation was in a test fold. The final score for each observation in the data set is then obtained by averaging confidence scores across the observation's ensemble. This procedure is repeated on the same set of data splits for each class weight $\beta = 0.7, 0.9, 0.99$ we consider.